Teamgeist: Die neuen Team-Development-Features in Apex 4.0

Holger Bär, science + computing AG

„Nicht noch ein Trackingsystem" wird sich der eine oder andere sicher denken. Im Projekt-Alltag ist es ja häufig so, dass man sich mit den verschiedensten vorhandenen Trackingsystemen auseinandersetzen muss: interne, externe beim Kunden und externe beim Lieferanten. Warum jetzt also noch einen draufsatteln und ein weiteres System etablieren? Ein sehr persönlicher Erfahrungsbericht über die Teamfunktionen in Apex 4.0.

Die erste Reaktion nach der erfolgreichen Installation von Apex 4.0 war nach dem Login in den ersten Workspace: „Ah, da gibt es einen neuen Reiter" und dann wurde dieser erst einmal kräftig ignoriert. Vier Wochen später führte die Frage an die Kollegen „Habt ihr euch das schon mal angeschaut?" nur zu einem desinteressierten Schulterzucken. Da antizyklisches Verhalten ja durchaus ein Erfolgsmodell ist, war das Interesse des Autors endgültig geweckt.

Der erste Eindruck ist für lange Zeit prägend in einer Beziehung – wie präsentieren sich also hier die Team-Funktionen? Auf den ersten Blick offenbart sich nicht viel Spektakuläres in der Oberfläche: ein paar modern gestaltete Symbole, die den Zugang zu Features, Milestones, To Dos, Bugs und Feedback eröffnen, ein paar leere Boxen zu den gleichen Punkten und Team Actions, die man an einer Hand abzählen kann. Immerhin wirkt es sehr aufgeräumt und übersichtlich. Wer dann nach unten scrollt, findet auch noch eine Tag „Cloud", deren Sinn sich durch einen beherzten Klick auf den Link zur (kontextsensitiven) Hilfe und etwas Handbuchstudium halbwegs erschließt. Auch wenn es nicht Liebe auf den ersten Blick ist, kann das auf jeden Fall ein angenehmer Abend mit der noch unbekannten Dame werden.

Viele Features

Bisher haben wir uns rein mit Äußerlichkeiten aufgehalten, jetzt sollten wir aber so langsam ins Gespräch kommen, also die ersten Daten eingeben. Die Reihenfolge der Symbole legt das Vorgehen nahe, erst mal Features einzugeben. Hier zeigt sich dann gleich das Konzept, das sich durch die gesamte Applikation zieht: sehr viele Informationen, die als Auswahlliste zur Verfügung gestellt werden (wie Feature Owner, Contributor, Focus Area, Release etc.), können bei Bedarf direkt neu eingegeben und müssen nicht erst in Form von Listen aufgebaut werden. Das ist an manchen Stellen dann wieder verwirrend, da eben nicht zum Beispiel die im Workspace definierten Benutzer zur Auswahl stehen, sondern auch hier frei eingetragen wird, wer was bearbeitet. Auch Releases werden völlig frei definiert, lediglich Meilensteine müssen angelegt sein, bevor man sie nutzen kann.

Abbildung 1: Die Toolbar im Team-Development

Schaut man noch etwas genauer hin, offenbaren sich Ansätze, die vermutlich erst in späteren Versionen wirklich richtig zum Tragen kommen: Ein Feature kann zum Beispiel veröffentlicht werden, aber auch die Hilfe schweigt sich dazu aus, wo die Veröffentlichung denn stattfinden soll und teilt lediglich mit, dass die Auswahl von „Yes" das Feature einem breiteren Publikum sichtbar macht (Select „Yes" to make this feature viewable to a broader audience).

Die Eingabe von geschätzten Stunden für die Erledigung des Features, so wie differenzierte Statusangaben und Bearbeiter zu User Interface, Testing, Documentation, Globalization, Security und Accessibility runden die Angaben zu Features ab, tauchen aber in kaum einem Report auf und sind damit von zweifelhaftem Wert.

Ein wenig irritierend: Da in einem Workspace ja mehrere Applikationen verwaltet werden können, ist die optionale Zuordnung eines Features zu einer Applikation als „Popup LOV" ausgeführt. Dieses wiederum ist in der Praxis etwas dick aufgetragen – ein normale Auswahlliste wäre in den meisten Anwendungsfällen sicher eher angebracht. Noch mehr reibt man sich verwundert die Augen, wenn man zum ersten Mal ein Feature auf „erledigt" setzt: das Enddatum darf und muss man trotzdem noch zusätzlich von Hand eingeben – Dynamic Actions haben hier noch nicht Einzug gehalten. Beliebige Tags, also Schlag- oder Stichworte können ebenfalls in einem extra Feld angegeben werden, diese füllen dann die bereits oben erwähnte Tag Cloud.

Die Angaben zu Milestones sind dagegen eher knapp aber ausreichend:

Name, Datum, Typ, Release und ob der Meilenstein für Features auswählbar ist, war schon fast alles. Umso überraschter ist man (beim ersten Anschauen), dass dann in der Ansicht zu Milestone Details ein Report „Top Remaining Feature Development Hours by Owner" auftaucht – die einzige Stelle, an der die Angaben zum Zeitbedarf derzeit aufzutauchen scheinen. Ob da einfach noch Platz war?

Meilensteine können überwacht und im Kalender betrachtet werden, sie dienen aber im Wesentlichen und naturgemäß dazu, den Fortschritt der Entwicklung im Verhältnis zur verbleibenden Zeit zu kontrollieren. Auch hier können wieder beliebige Schlagworte als Tags vergeben werden.

To do or not to do

Die To dos weisen auch keine Überraschungen auf; es können alle relevanten Angaben gemacht werden, um eine Aufgabe zu beschreiben und um Bezüge zu Applikation (ebenfalls über Popup LOV auszuwählen), Features und Milestones herzustellen. Weiterhin kann der Fortschritt der Bearbeitung sowohl prozentual über eine Auswahlbox als auch in Textform im Progress Log dokumentiert werden. Eine Darlegung des geschätzten Aufwands wirkt sich anscheinend nirgends aus, die Angabe tatsächlich verbrauchter Zeit fehlt vollständig. Die zugehörigen Reporting-Möglichkeiten in Form von Dashboard, Calendar und Progress Log offenbaren wenig Neues, zeigen aber (bei Features, Milestone und auf der Startseite) wie man mit Apex gut aussehende und vor allem übersichtlich aufgebaute Reportseiten designen kann. Natürlich darf man auch hier nach Herzenslust taggen.

Abbildung 2: Entgegen der Aufschrift muss der Entwickler hier nach wie vor selbst tätig werden

Mein Bug – dein Bug

Als letzte Kategorie für selbsterfasste Daten finden sich noch Bugs, denen man ebenfalls die wichtigsten Informationen zur Beschreibung mitgeben kann. Die größte Überraschung auf dieser Seite ist die Tatsache, dass die Auswahl der Applikation hier im Gegensatz zu den vorherigen nun wunschgemäß als normale Listbox ausgeführt ist. Eher irritierend ist dagegen, dass man in der Angabe des Context nur auf bereits existierende Features und – besonders störend – auf To dos zurückgreifen kann, was man dann doch andersherum erwarten würde: Ein Bug löst ein To do aus. Eine etwas unfreiwillige Komik geht mit der Aufforderung „Create Bug" einher, aber diese Arbeit bleibt auch weiterhin dem Programmierer überlassen.

Kontaktfreudig

Eine sehr spannende Funktion ist die Möglichkeit, in die Applikation eine

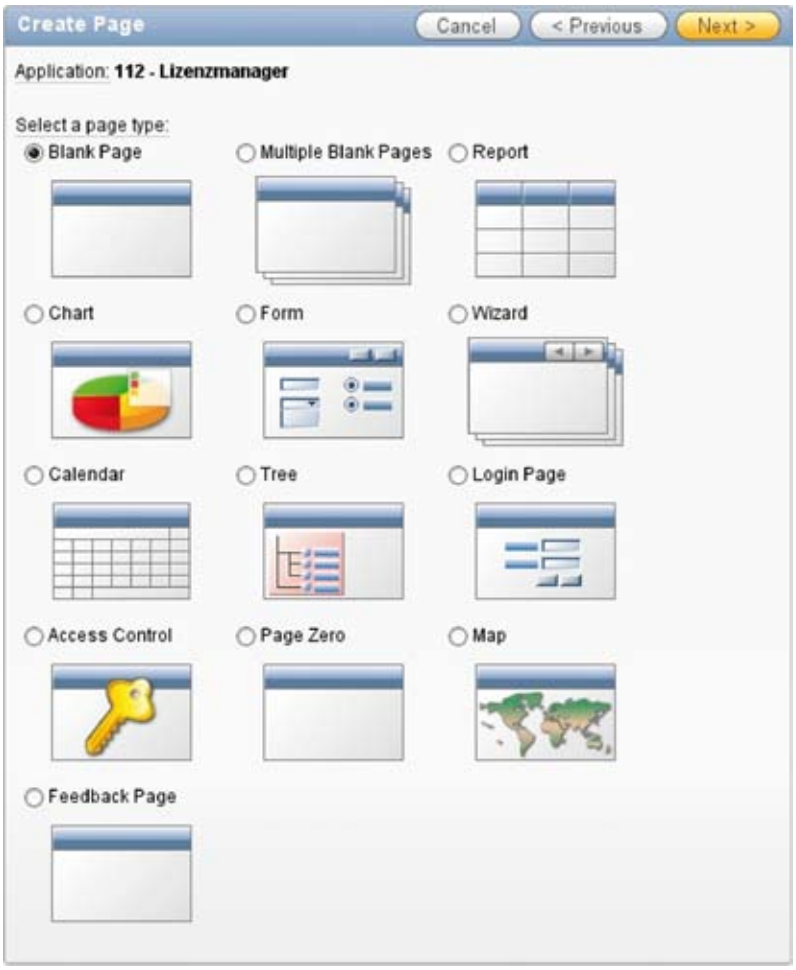

Abbildung 3: Anlegen einer Feedback-Seite über den Create Page Wizard

Feedbackseite einzubauen. Enthält das Seiten-Template den Anker „#NAVIGATION_BAR#“ und wurde in den Applikations-Eigenschaften „Feedback“ erlaubt, wird automatisch ein Link in der Applikation eingeblendet, über den die – als eigener Typ im Seite-Erzeugen-Wizard auswählbare – Feedbackseite als Popup aufgerufen wird.

Auf diese Weise können Endanwender sehr einfach ihr Feedback in der Applikation eingeben. Feedback enthält neben den vom Anwender eingetragenen Angaben standardmäßig Informationen zu der Seite, aus der der Link aufgerufen wurde, den Benutzernamen, das Betriebssystem und die Browser-Version. Das Feedback steht dann zur Weiterverarbeitung zur Verfügung und lässt sich sogar exportieren, um Daten aus einem Produktivsystem wieder in die Entwicklung zurückzuführen. Ex- und Import für Feedback sind übrigens über den Application Builder im „Export/Import“-Assistent zu finden – eine zwar leicht nachvollziehbare, aber vielleicht doch nur bedingt sinnvolle Entscheidung, da der Bezug zur Applikation (im Vergleich zu den Seiten des Team-Developments) doch eher gering ist.

Aus Feedback können per Knopfdruck ein Feature, ein To do oder ein Bug erzeugt werden – hier verwundert, dass die im Feedback aufgeführten Informationen wie Browser und Betriebssystem nicht in die entsprechenden Felder einer Bugbeschreibung übernommen werden.

Auf Feedback kann man mit einem Follow-up antworten. Wer jetzt aber meint, dass damit vielleicht sogar ein automatischer Mailversand einhergeht, der hat sich da leider zu viel erhofft – zu mehr als einer Anzeige für Entwickler auf der Feedback-Übersicht reicht es vorerst nicht. Immerhin sind die Daten recht einfach über die Views „apex_team_feedback“ und „apex_team_feedback_followup“ zugänglich, sodass eine entsprechende Anzeige in der Applikation möglich ist. Der Export des Feedbacks aus der Entwicklungsumgebung ermöglicht auch die Auswahl des Zielsystems, in das die Daten eingespielt werden sollen, damit die Antworten zumindest ihr Ziel finden können – alles in allem aber ein eher aufwändiger manueller Prozess. Dennoch hat das Vorgehen auch seine Vorteile, da man die Dokumentation in dem Entwicklungssystem hält, und nur die öffentlichen Kommentare sowie direkte Follow-ups wieder in die Produktionsumgebung zurückgespielt werden – auch wenn diese vielleicht im Kundennetz steht. Ebenso erlaubt dieses Vorgehen das „Einer für alle“-Modell (ein Workspace für alle Applikationen), ohne dass man den Zusammenhang von Feedback und zugehöriger Kundenapplikation verliert.

Randerscheinung

Wie zu Beginn bereits kurz erwähnt, gibt es noch sogenannte „Team Actions“. Dahinter verbergen sich Pflege-Funktionen, um beispielsweise nach personellen Änderungen im Entwicklungsteam alle zugewiesenen Aufgaben eines Entwicklers an den Ersatz oder die verbleibenden Entwickler umzuverteilen oder bei Features beziehungsweise Bugs gesammelt den Zieltermin um eine beliebige Anzahl Tage nach hinten zu verschieben.

Eher in den Bereich der Technologie-Studie fallen auch die Möglichkeiten, Links auf externe URLs aufzunehmen oder einen Newsticker zu befüllen – grundsätzlich in manchen Szenarien bestimmt hilfreich, aber Dokumentation und Kommunikation werden im Allgemeinen (hoffentlich) schon besser gelöst sein.

Für wen es sich lohnt

Es stellt sich natürlich die Frage, für wen der Einsatz der Team-Development-Features sinnvoll ist. Die einfache und kurze Antwort ist: Alle (Apex-) Entwicklungsteams, die noch über keinerlei Software-Unterstützung für den

Entwicklungsprozess verfügen, werden vom Einsatz der Team-Development-Features profitieren – die Integration der Management-Funktionalitäten in die Entwicklungsumgebung überzeugt einfach. Die Dokumentation von Anforderungen und Entwicklungsfortschritt innerhalb der gleichen Oberfläche wie deren Umsetzung erleichtern die Verwendung sehr. Und zumindest am Anfang ist auch der Ansporn deutlich spürbar, die ansprechenden Dashboards mit möglichst vielen „fast fertig"-Anzeigen zu füllen. Das Arbeiten im Team funktioniert ohne großen administrativen Overhead, da keine User anzulegen sind und diese sich nicht in ein zusätzliches Tool einarbeiten müssen.

Schwieriger wird die Entscheidung, wenn bereits Tools wie das in der Java-Welt sehr beliebte JIRA etabliert sind. Ein Parallel-Betrieb für das gleiche Projekt lohnt sich in den wenigsten Fällen und man müsste sich die entsprechenden Export-/ Import-Schnittstellen erst von Hand generieren, um nicht alle Daten mehrfach pflegen zu müssen.

Die wenigen Schwächen der neuen Features fallen nur gering ins Gewicht. Für umfangreichere Projekte würde man sich vielleicht mehr Reporting-Möglichkeiten in Bezug auf Terminschienen etwa in Form eines Gantt-Diagramms, Ressourcenbedarf und -verbrauch sowie einen Ausbau der Feedback-Funktionalität wünschen. Man darf gespannt sein, was künftige Apex-Versionen bieten werden. Die Tag „Cloud" scheint eher eine Technologie-Studie denn ein wirklich sinnvolles Tool zu sein, aber das ist vielleicht auch Geschmackssache.

Am meisten vermisst man die Möglichkeit, automatisch per E-Mail über neue Eintragungen informiert zu werden, egal ob Feature, Bug oder Feedback. Die Feedbackfunktionen allein haben das Team des Autors jedenfalls für die gerade in der Entwicklung befindliche Applikation zum Umstieg bewogen, auch wenn vorläufig mangels standardisierter Darstellung mit Apex-Mitteln nur die Übertragung in das Entwicklungssystem genutzt wird. Ein Gantt-Diagramm ist ja in Apex 4.0 zur Not auch mit ein wenig SQL schnell erstellt (Siehe Listing 1 und Abbildung 5).

Abbildung 4: Feedback aus Entwicklung und Produktion

Abbildung 5: Ein Gantt sagt mehr als die Kalenderdarstellungen

```
select null,
       feature_name,
       feature_friendly_id,
       null parent_id,
       nvl( start_date,
            to_
date(,01.09.2010‘,‘dd.mm.yyyy‘))
start_date,   due_date end_date,
       feature_status progress
from apex_team_features
where applikation_id=v(,APP_ID‘)
order by feature_friendly_id;
```

Listing 1: Mit wenig Code zum ersten Gantt-Diagramm

Kontakt:
Holger Bär
holger.baer@science-computing.de

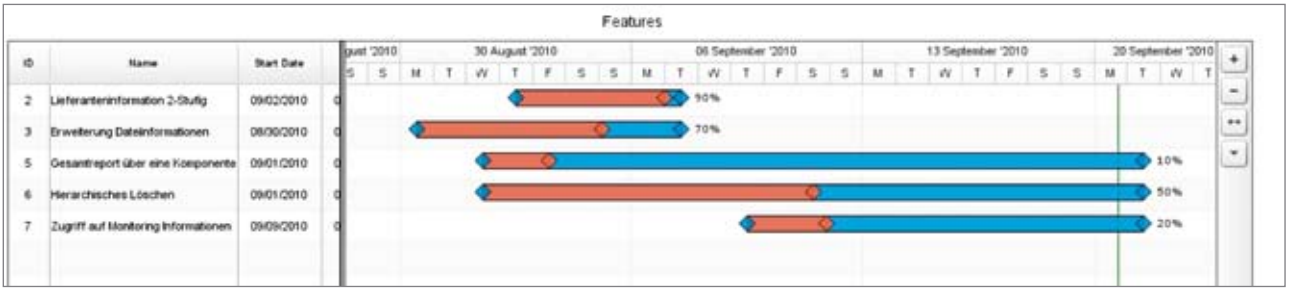

Abbildung 6: Ein Gantt